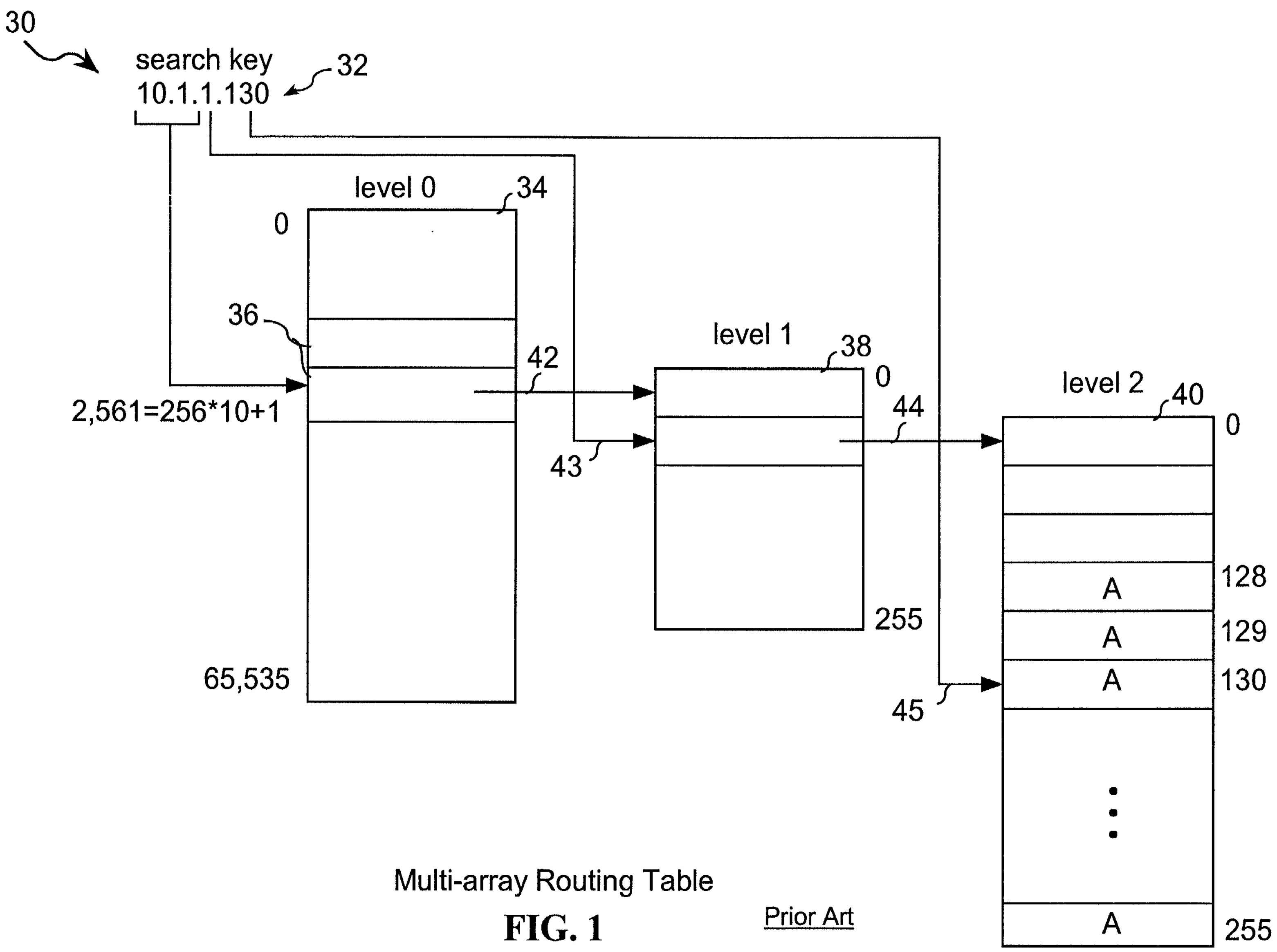


Multi-array Routing Table

FIG. 1

Prior Art

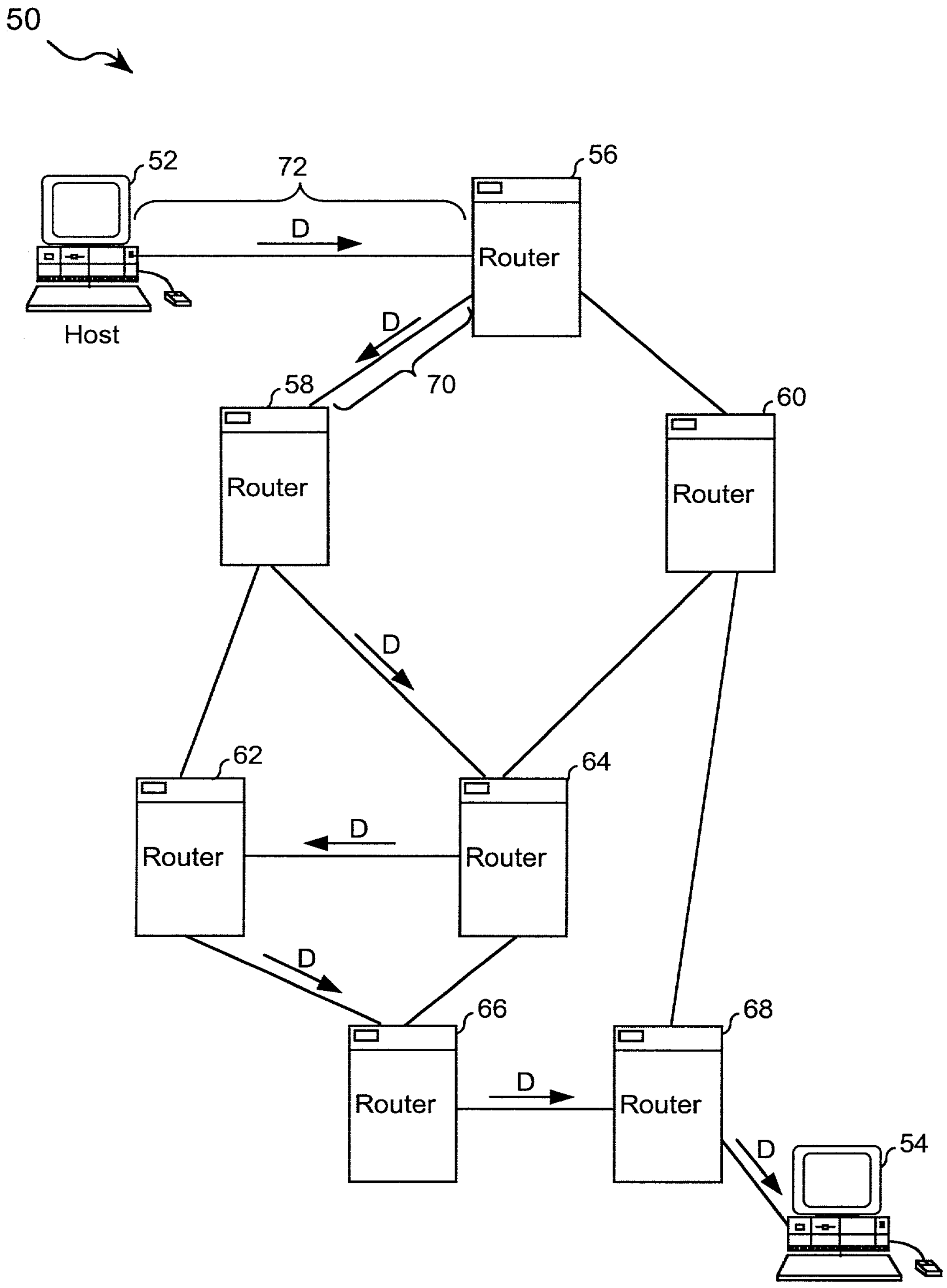
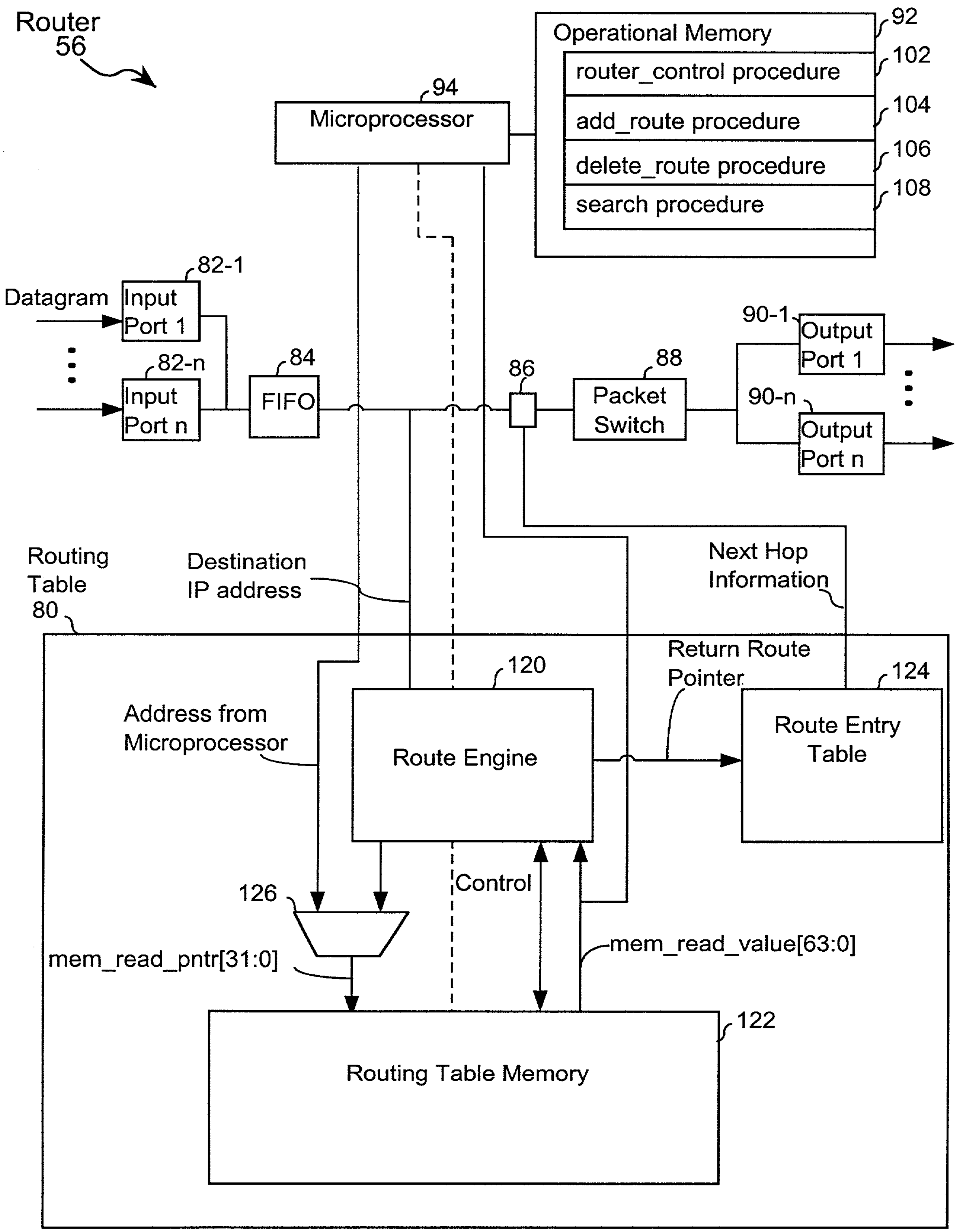

50
52
72
56
D
Router
Host
D
70
58
60
Router
Router
D
62
64
D
Router
Router
D
66
68
D
Router
Router
D
54

FIG. 2


Router
56
Operational Memory
92
router_control procedure
102
add_route procedure
104
delete_route procedure
106
search procedure
108
Microprocessor
94
Datagram
Input Port 1
82-1
Input Port n
82-n
FIFO
84
86
Packet Switch
88
90-1
Output Port 1
90-n
Output Port n
Routing Table
80
Destination IP address
Next Hop Information
Return Route Pointer
Route Engine
120
Route Entry Table
124
Address from Microprocessor
126
Control
mem_read_pntr[31:0]
mem_read_value[63:0]
Routing Table Memory
122

Router

FIG. 3

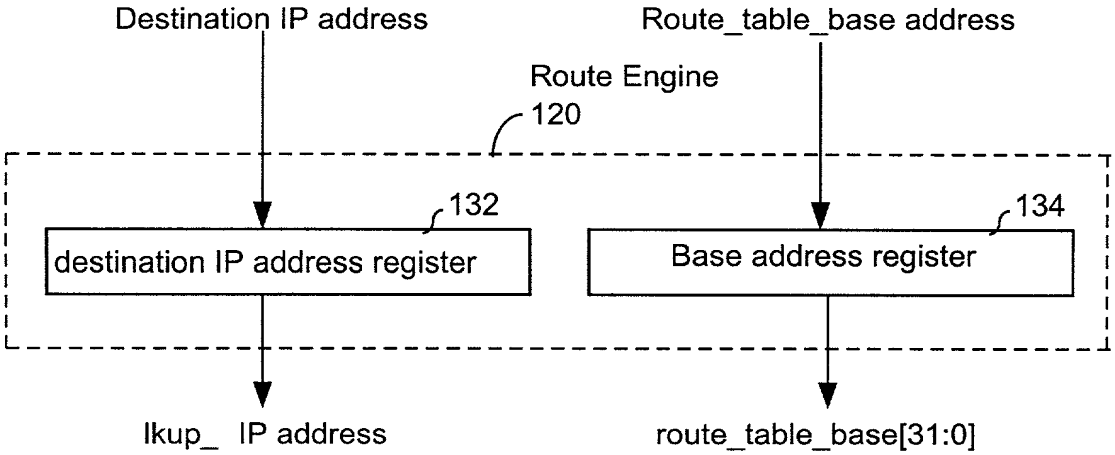

Destination IP address
Route_table_base address
Route Engine
120
132
134
destination IP address register
Base address register
lkup_ IP address
route_table_base[31:0]

FIG. 4

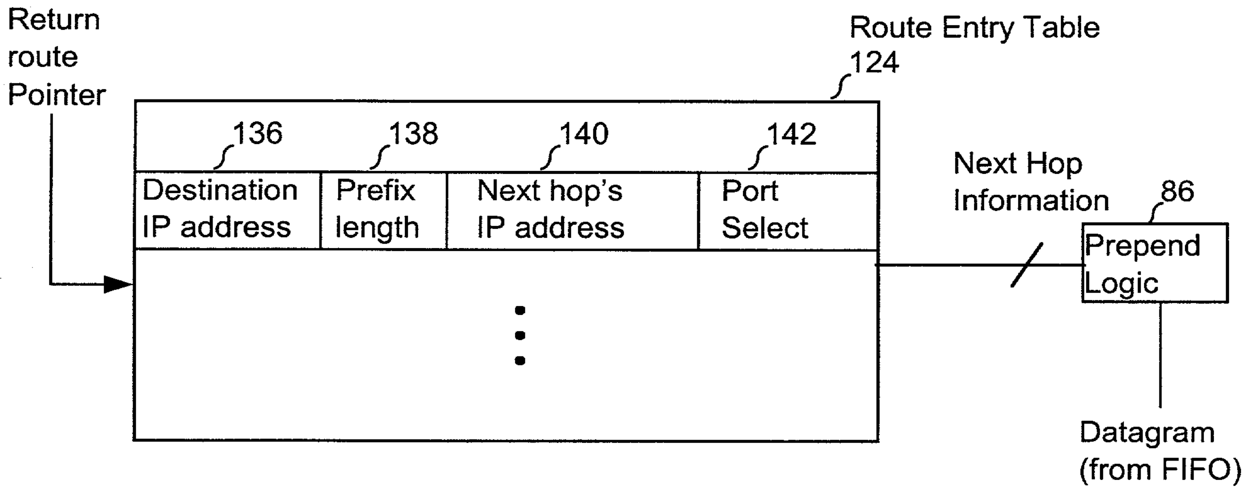

Return
route
Pointer
Route Entry Table
124
136
138
140
142
Destination
IP address
Prefix
length
Next hop's
IP address
Port
Select
Next Hop
Information
86
Prepend
Logic
Datagram
(from FIFO)

FIG. 5

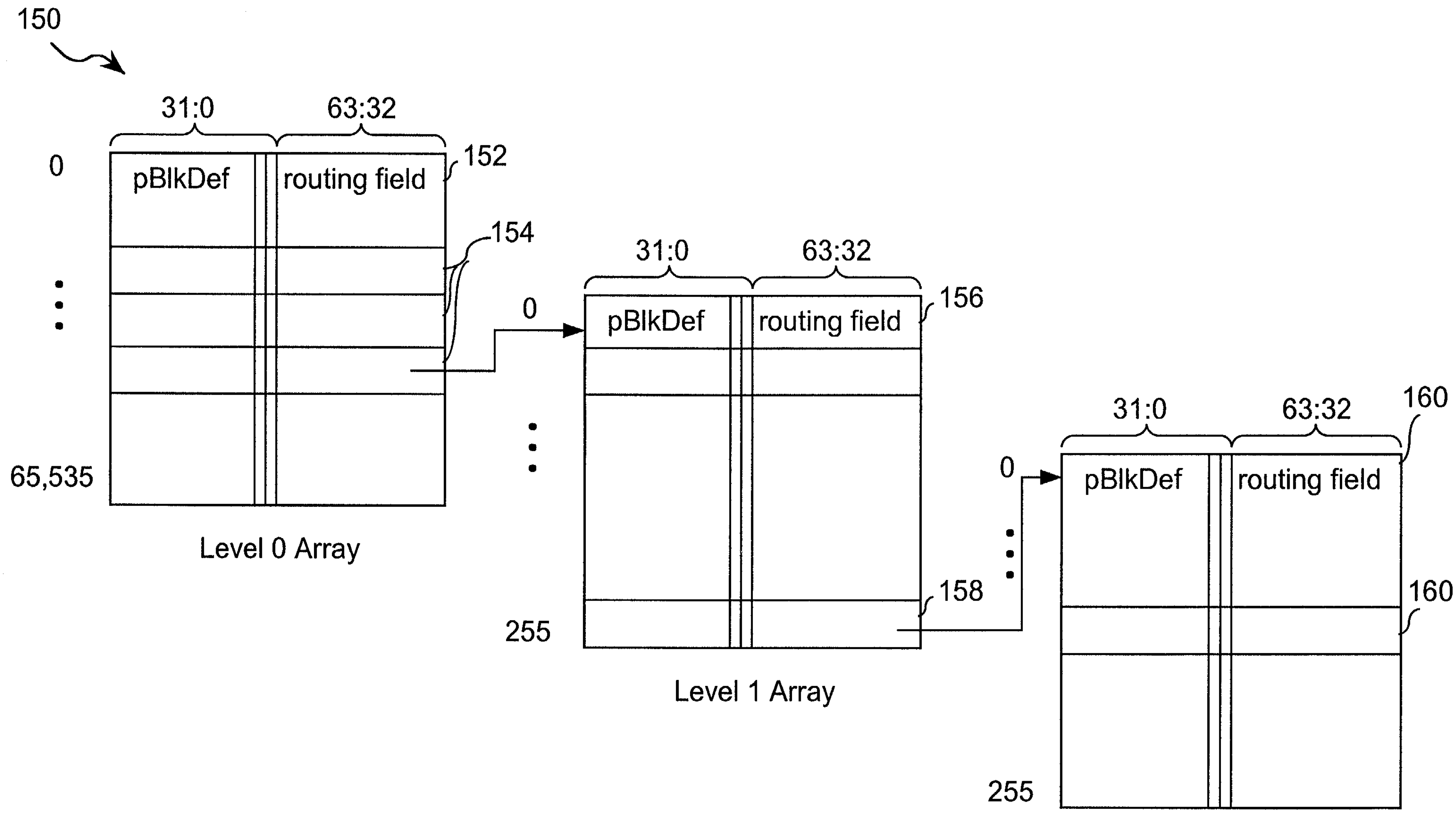

150
31:0
63:32
0
pBlkDef
routing field
152
154
65,535
Level 0 Array
31:0
63:32
0
pBlkDef
routing field
156
158
255
Level 1 Array
31:0
63:32
160
0
pBlkDef
routing field
160
255
Level 2 Array

FIG. 6

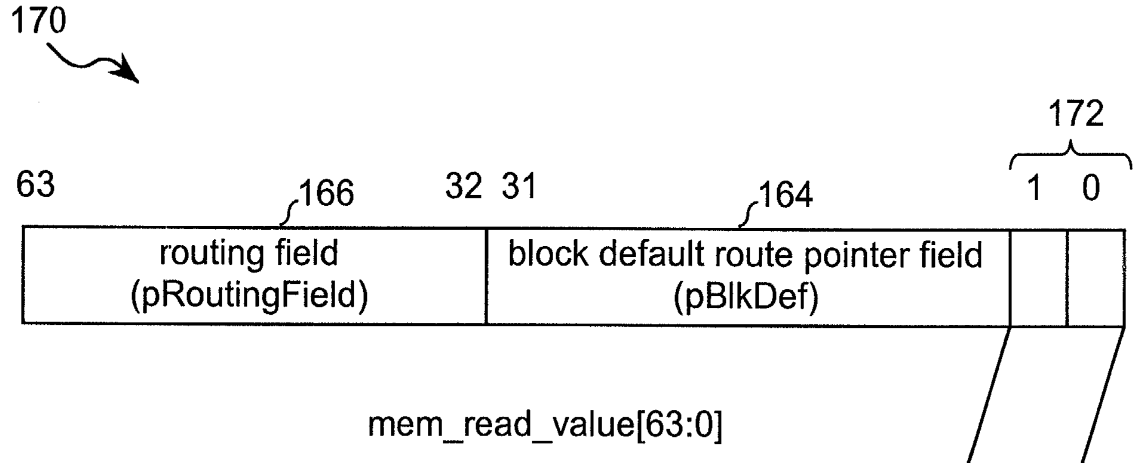
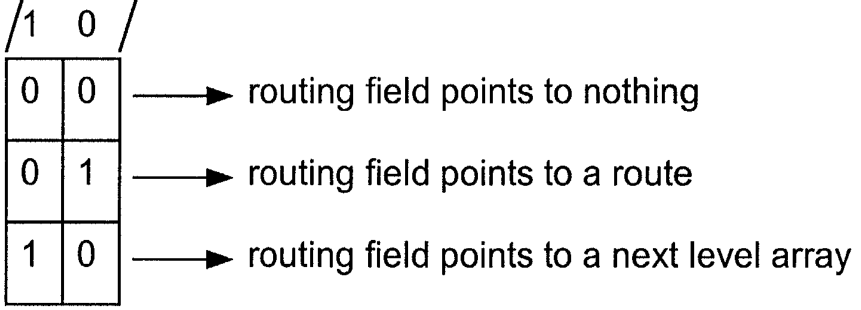
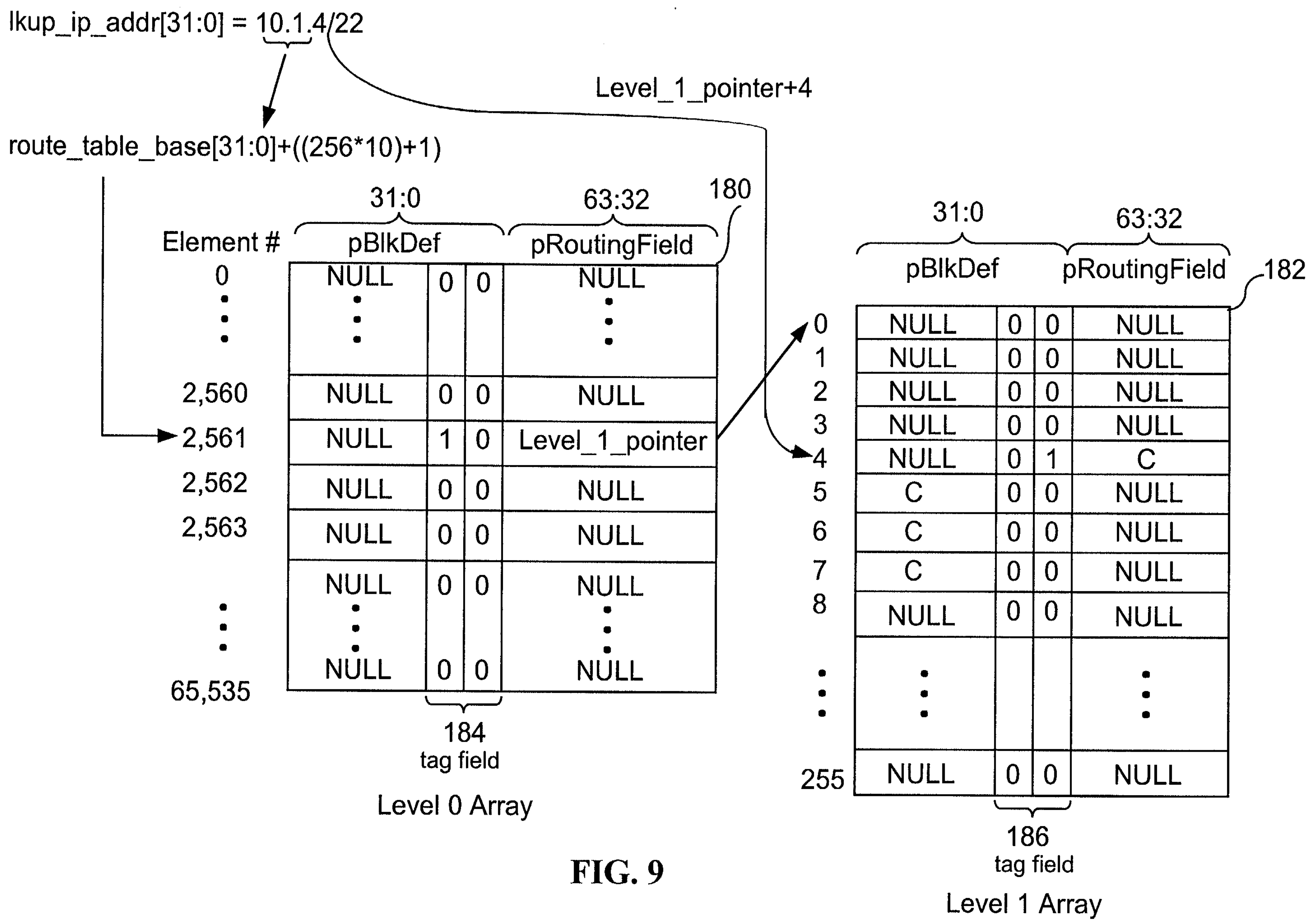


FIG. 7



1	0	
0	0	routing field points to nothing
0	1	routing field points to a route
1	0	routing field points to a next level array

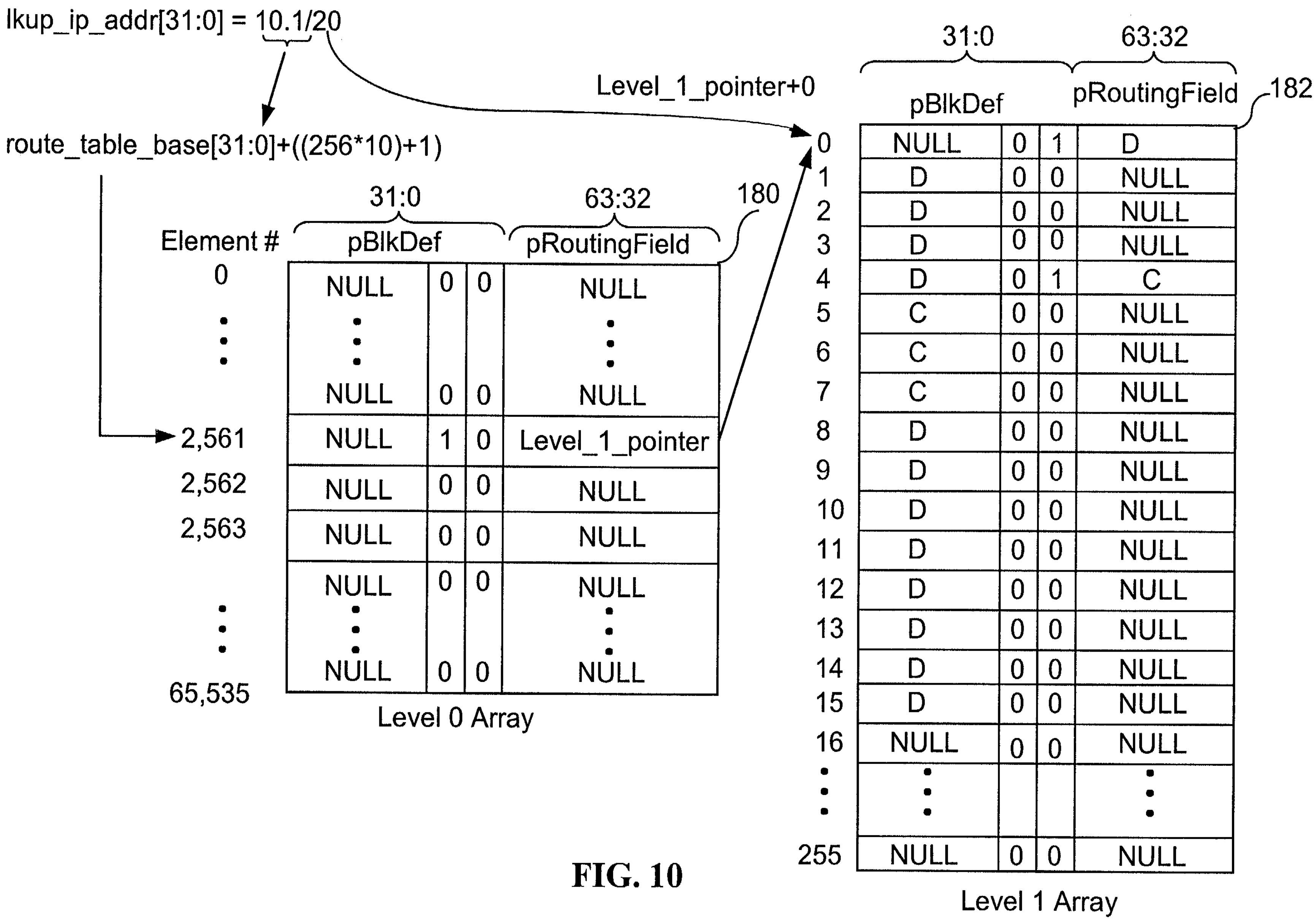
FIG. 8



Element #	pBlkDef			pRoutingField
0 ⋮	NULL ⋮	0	0	NULL ⋮
2,560	NULL	0	0	NULL
2,561	NULL	1	0	Level_1_pointer
2,562	NULL	0	0	NULL
2,563	NULL	0	0	NULL
⋮ 65,535	NULL ⋮ NULL	0 0	0 0	NULL ⋮ NULL

	pBlkDef			pRoutingField
0	NULL	0	0	NULL
1	NULL	0	0	NULL
2	NULL	0	0	NULL
3	NULL	0	0	NULL
4	NULL	0	1	C
5	C	0	0	NULL
6	C	0	0	NULL
7	C	0	0	NULL
8	NULL	0	0	NULL
⋮	⋮			⋮
255	NULL	0	0	NULL

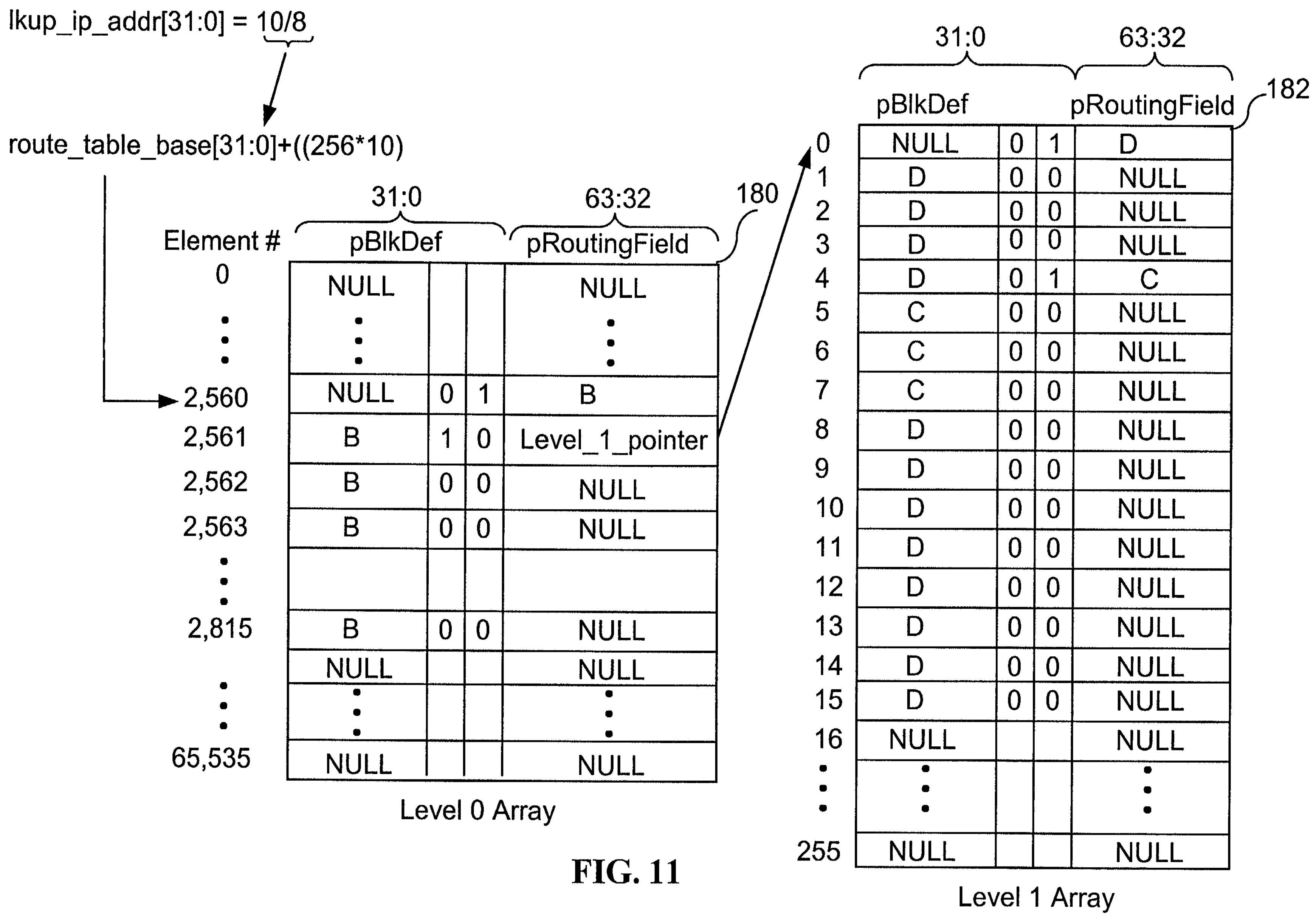
FIG. 9


lkup_ip_addr[31:0] = 10.1/20
route_table_base[31:0]+((256*10)+1)
Level_1_pointer+0
180
182
31:0
63:32
Level 0 Array
Level 1 Array

Element #	pBlkDef (31:0)			pRoutingField (63:32)
0	NULL	0	0	NULL
⋮	⋮			⋮
	NULL	0	0	NULL
2,561	NULL	1	0	Level_1_pointer
2,562	NULL	0	0	NULL
2,563	NULL	0	0	NULL
	NULL	0	0	NULL
⋮	⋮			⋮
65,535	NULL	0	0	NULL

	pBlkDef (31:0)			pRoutingField (63:32)
0	NULL	0	1	D
1	D	0	0	NULL
2	D	0	0	NULL
3	D	0	0	NULL
4	D	0	1	C
5	C	0	0	NULL
6	C	0	0	NULL
7	C	0	0	NULL
8	D	0	0	NULL
9	D	0	0	NULL
10	D	0	0	NULL
11	D	0	0	NULL
12	D	0	0	NULL
13	D	0	0	NULL
14	D	0	0	NULL
15	D	0	0	NULL
16	NULL	0	0	NULL
⋮	⋮			⋮
255	NULL	0	0	NULL

FIG. 10



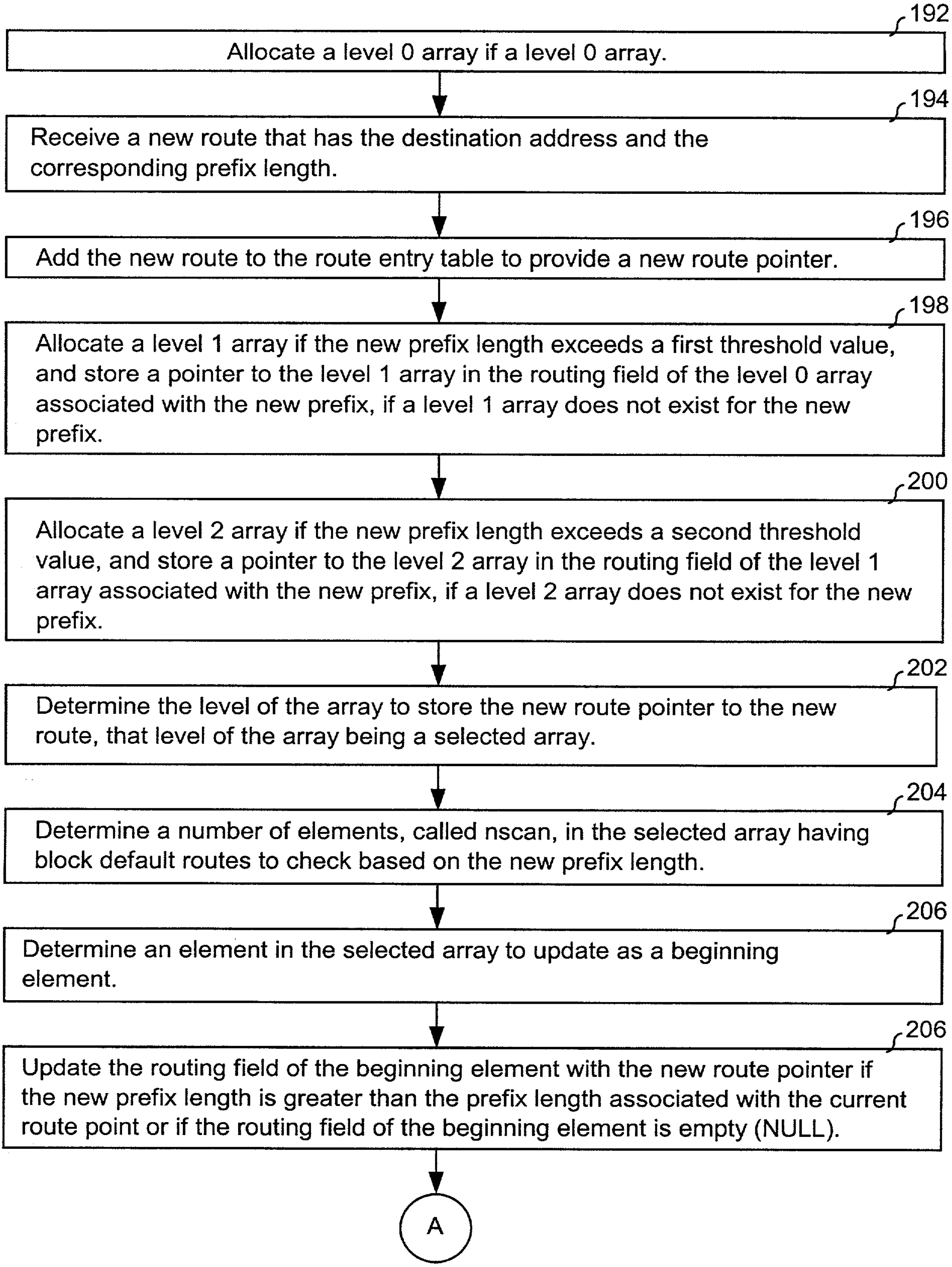
Element #	pBlkDef (31:0)			pRoutingField (63:32)
0	NULL			NULL
⋮	⋮			⋮
2,560	NULL	0	1	B
2,561	B	1	0	Level_1_pointer
2,562	B	0	0	NULL
2,563	B	0	0	NULL
⋮				
2,815	B	0	0	NULL
	NULL			NULL
⋮	⋮			⋮
65,535	NULL			NULL

Level 0 Array

	pBlkDef (31:0)			pRoutingField (63:32)
0	NULL	0	1	D
1	D	0	0	NULL
2	D	0	0	NULL
3	D	0	0	NULL
4	D	0	1	C
5	C	0	0	NULL
6	C	0	0	NULL
7	C	0	0	NULL
8	D	0	0	NULL
9	D	0	0	NULL
10	D	0	0	NULL
11	D	0	0	NULL
12	D	0	0	NULL
13	D	0	0	NULL
14	D	0	0	NULL
15	D	0	0	NULL
16	NULL			NULL
⋮	⋮			⋮
255	NULL			NULL

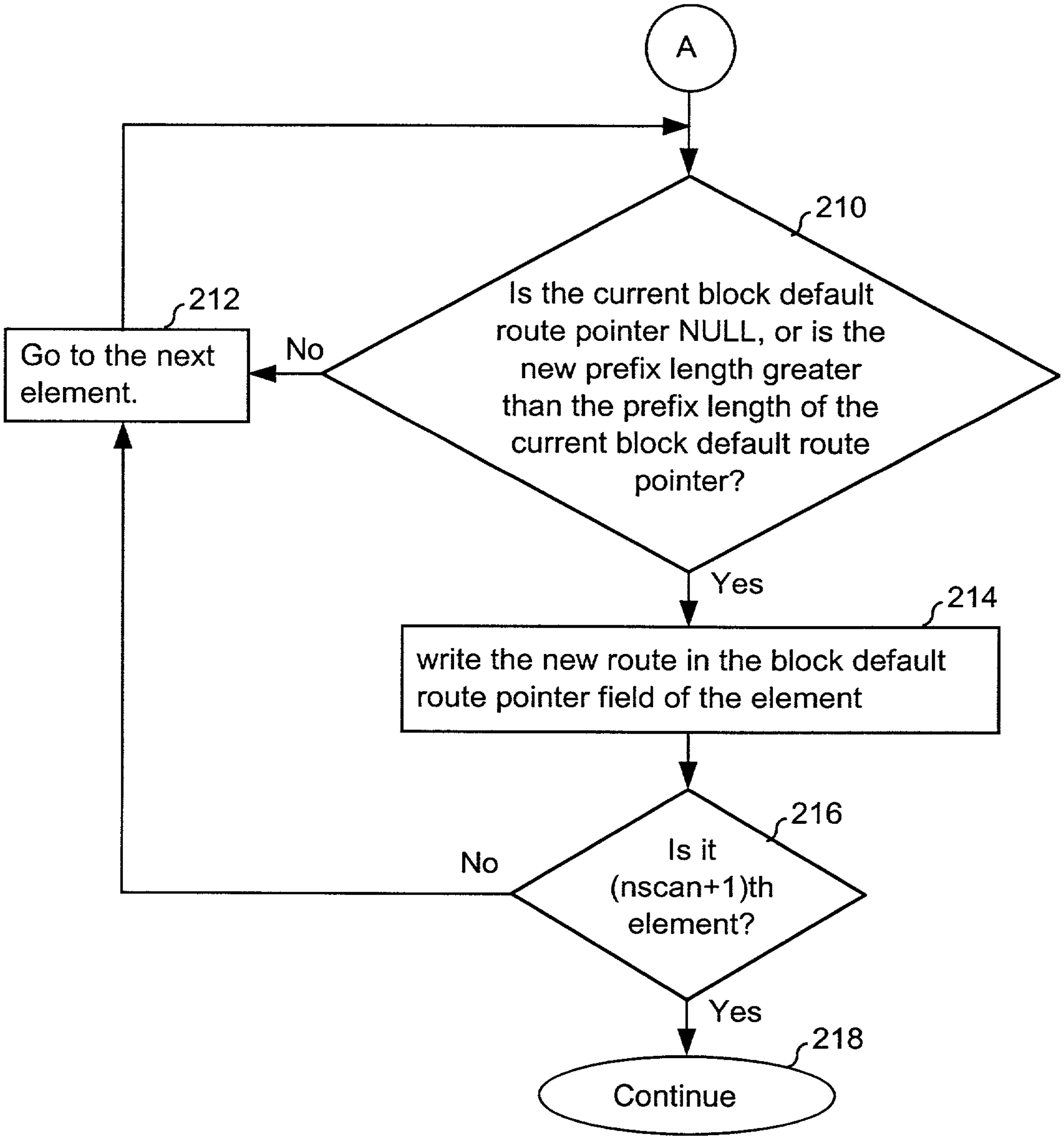
Level 1 Array

FIG. 11


192
Allocate a level 0 array if a level 0 array.
194
Receive a new route that has the destination address and the corresponding prefix length.
196
Add the new route to the route entry table to provide a new route pointer.
198
Allocate a level 1 array if the new prefix length exceeds a first threshold value, and store a pointer to the level 1 array in the routing field of the level 0 array associated with the new prefix, if a level 1 array does not exist for the new prefix.
200
Allocate a level 2 array if the new prefix length exceeds a second threshold value, and store a pointer to the level 2 array in the routing field of the level 1 array associated with the new prefix, if a level 2 array does not exist for the new prefix.
202
Determine the level of the array to store the new route pointer to the new route, that level of the array being a selected array.
204
Determine a number of elements, called nscan, in the selected array having block default routes to check based on the new prefix length.
206
Determine an element in the selected array to update as a beginning element.
206
Update the routing field of the beginning element with the new route pointer if the new prefix length is greater than the prefix length associated with the current route point or if the routing field of the beginning element is empty (NULL).
A

Flowchart for Adding a Route

FIG. 12A


A
210
Is the current block default route pointer NULL, or is the new prefix length greater than the prefix length of the current block default route pointer?
No
212
Go to the next element.
Yes
214
write the new route in the block default route pointer field of the element
216
Is it (nscan+1)th element?
No
Yes
218
Continue

Flowchart for Adding a Route (continued)

FIG. 12B

```
addRoute(ipa, plen)
/* ipa: destination address of new route,
   plen: prefix length of new route */

Level 0, level 1 and level 2 arrays are allocated and next level route pointers in the
level 0 and level 1 routes are updated.

        array[] = the array to which the new route is added
        begin   = getIndex(ipa,plen)  /* determine which array and element of that
                                         array stores the new route pointer as a route */
        nScan   = getNscan(plen)      /* determine a number of elements to scan */

        /* Update the array with the new route */
        array[begin].pRoutingField = pointer to the new route

        /* Update the block default route */
        i = begin + 1                 /* i points to the next element */
        While nScan-- > 0
                If plen > prefix length of the route pointed by array[i].pBlkdef then
                        array[i].pBlkDef = pointer to the new route
                i = i + 1
```

Pseudo code for adding a route

FIG. 13

Memory Write Sequence for Route D Inserted
in Level 1 Array

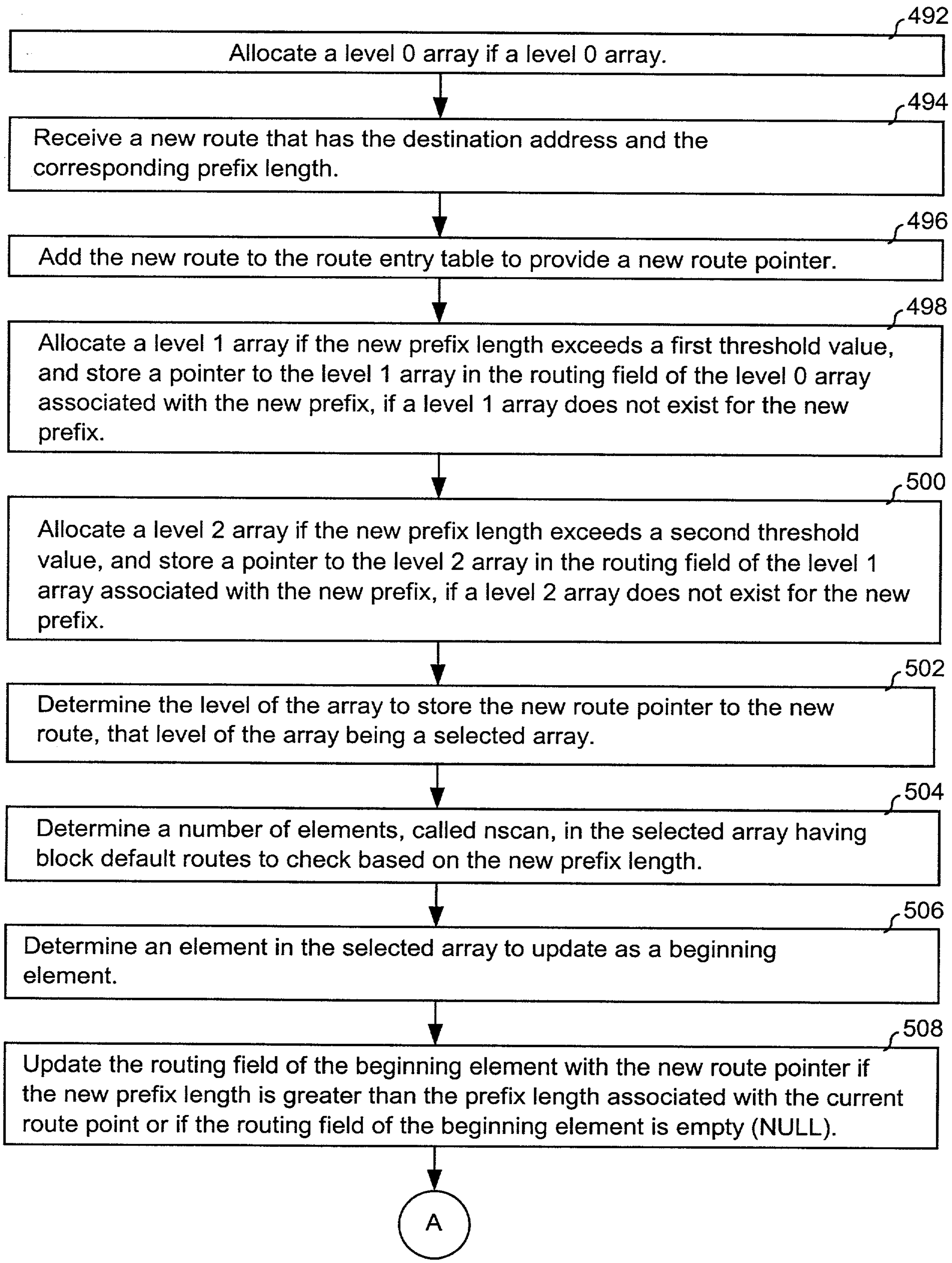
Element 0	Write D in Routing Field
Element 1	Write D in Default Route
Element 2	Write D in Default Route
Element 3	Write D in Default Route
Element 4	Write D in Default Route
Element 8	Write D in Default Route
Element 9	Write D in Default Route
Element 10	Write D in Default Route
Element 11	Write D in Default Route
⋮	⋮

FIG. 14

Memory Write Sequence for Route D Deleted
from Level 1 Array

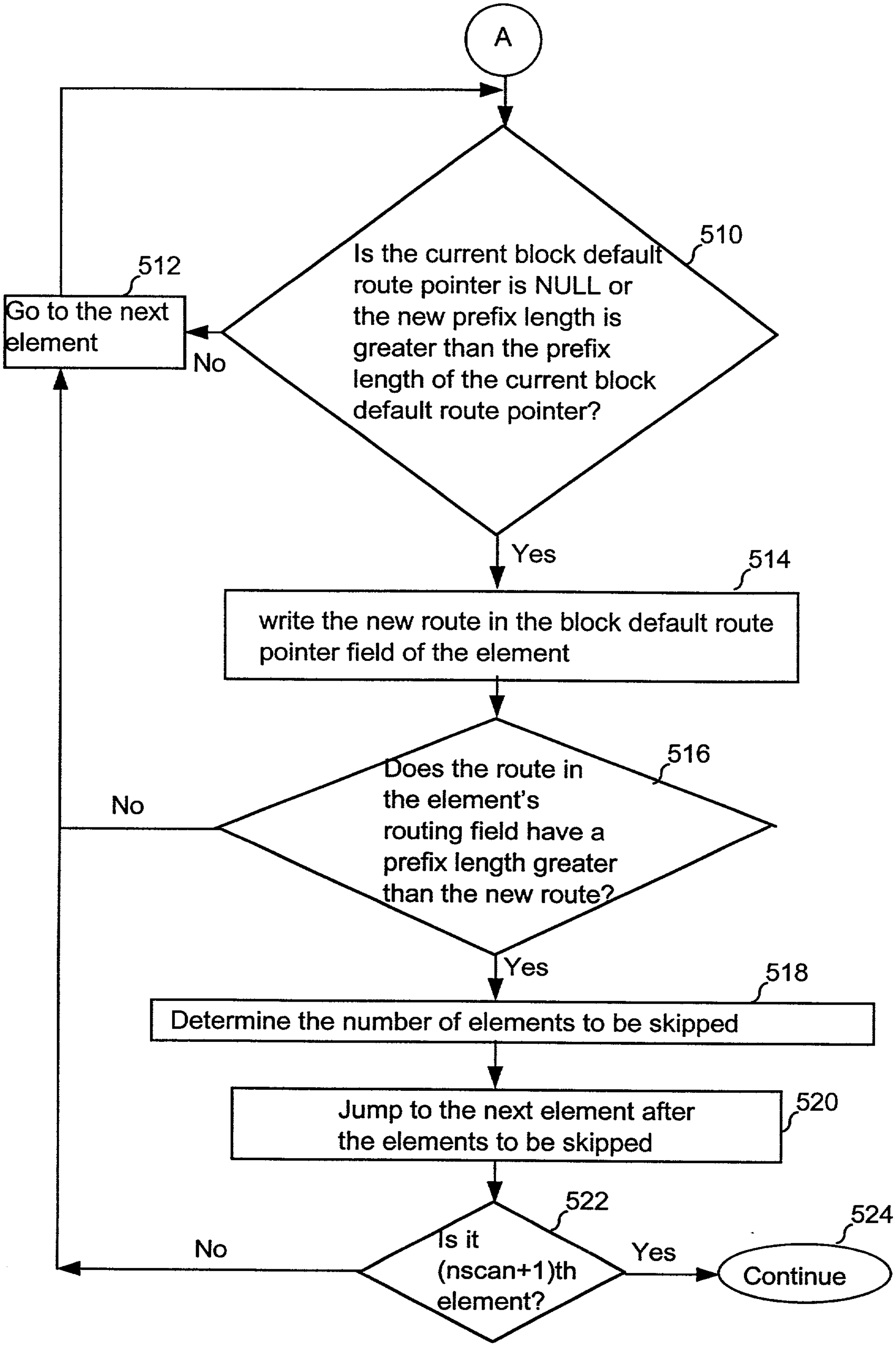
Element 0	Delete D from Routing Field
Element 1	Delete D from Default Route
Element 2	Delete D from Default Route
Element 3	Delete D from Default Route
Element 4	Delete D from Default Route
Element 8	Delete D from Default Route
Element 9	Delete D from Default Route
Element 10	Delete D from Default Route
Element 11	Delete D from Default Route
⋮	⋮

FIG. 17


492
Allocate a level 0 array if a level 0 array.
494
Receive a new route that has the destination address and the corresponding prefix length.
496
Add the new route to the route entry table to provide a new route pointer.
498
Allocate a level 1 array if the new prefix length exceeds a first threshold value, and store a pointer to the level 1 array in the routing field of the level 0 array associated with the new prefix, if a level 1 array does not exist for the new prefix.
500
Allocate a level 2 array if the new prefix length exceeds a second threshold value, and store a pointer to the level 2 array in the routing field of the level 1 array associated with the new prefix, if a level 2 array does not exist for the new prefix.
502
Determine the level of the array to store the new route pointer to the new route, that level of the array being a selected array.
504
Determine a number of elements, called nscan, in the selected array having block default routes to check based on the new prefix length.
506
Determine an element in the selected array to update as a beginning element.
508
Update the routing field of the beginning element with the new route pointer if the new prefix length is greater than the prefix length associated with the current route point or if the routing field of the beginning element is empty (NULL).
A

Flowchart for Adding a Route with Automatic Skipping

FIG. 15A


A
Is the current block default route pointer is NULL or the new prefix length is greater than the prefix length of the current block default route pointer?
510
512
Go to the next element
No
Yes
514
write the new route in the block default route pointer field of the element
516
Does the route in the element's routing field have a prefix length greater than the new route?
No
Yes
518
Determine the number of elements to be skipped
520
Jump to the next element after the elements to be skipped
522
Is it (nscan+1)th element?
No
Yes
524
Continue

Flowchart for Adding a Route with Autmatic Skipping (continued)

FIG. 15B

```
addRoute(ipa, plen)
/* ipa: destination address of new route,
   plen: prefix length of new route */

Level 0, level 1 and level 2 arrays are allocated and next level route pointers in the
level 0 and level 1 routes are updated.

        array[] = the array to which the new route is added
        begin   = getIndex(ipa,plen) /* determine which array and element of that
                                        array stores the new route pointer as a route */
        nScan  = getNscan(plen)      /* determine a number of elements to scan */

        /* Update the array with the new route */
        array[begin].pRoutingField = pointer to the new route

        /* Update the block default route */
        i = begin + 1                   /* i points to the next element */
        While nScan-- > 0
                If plen > prefix length of the route pointed by array[i].pBlkdef then
                        array[i].pBlkDef = pointer to the new route
                i = i + 1
        If array[i].pRoutingField = Not Null then
                nSkip = getNscan (plen) + 1        /* get number of elements
                                                      to be skipped */
        i = i + nSkip           /* jump to the next element after skipping */
```

Pseudo Code for Enhanced Route Addition with Automatic Skipping

FIG. 16

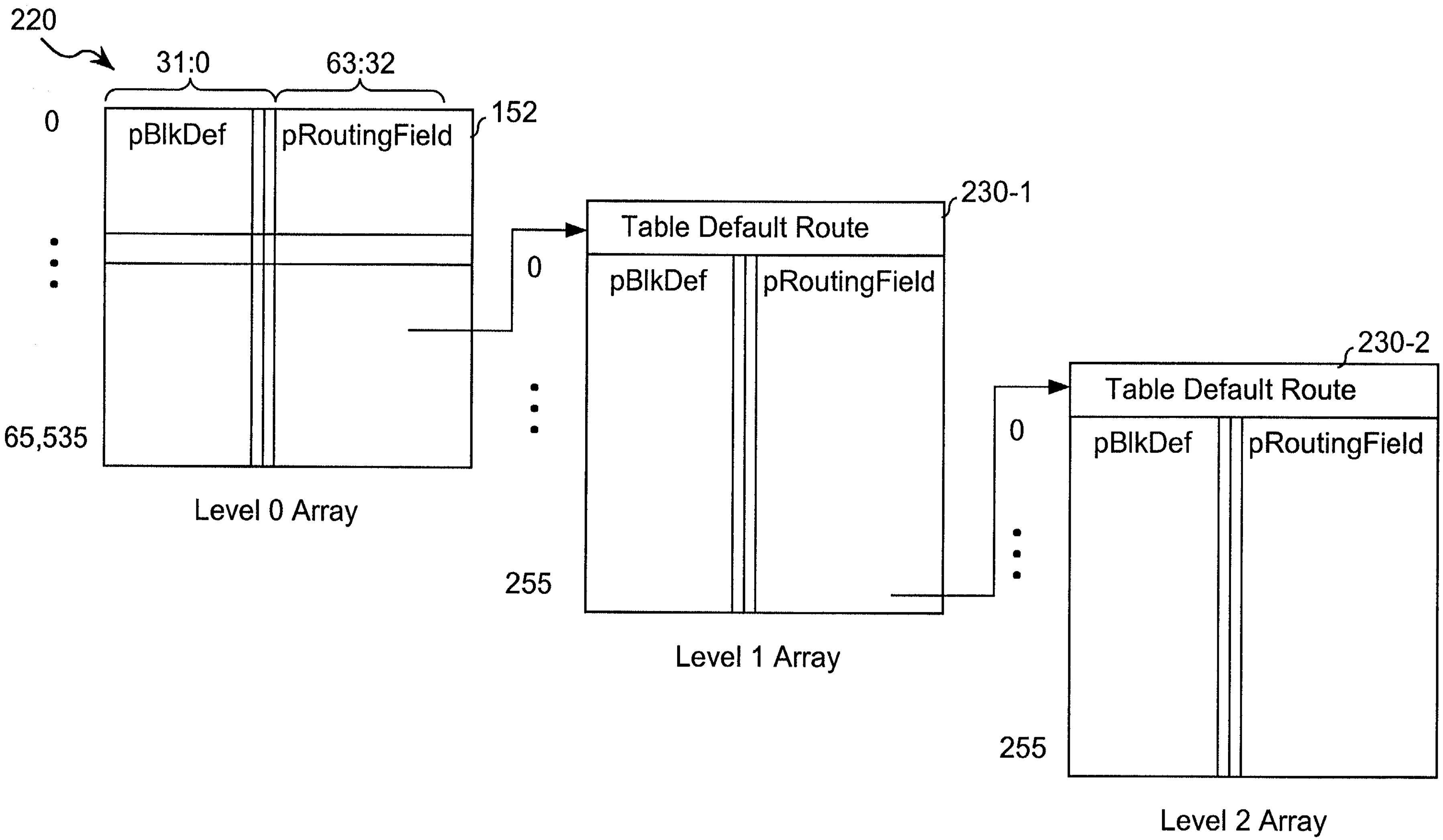

220
31:0
63:32
152
0
65,535
pBlkDef
pRoutingField
Level 0 Array
230-1
Table Default Route
pBlkDef
pRoutingField
0
255
Level 1 Array
230-2
Table Default Route
pBlkDef
pRoutingField
0
255
Level 2 Array

FIG. 18

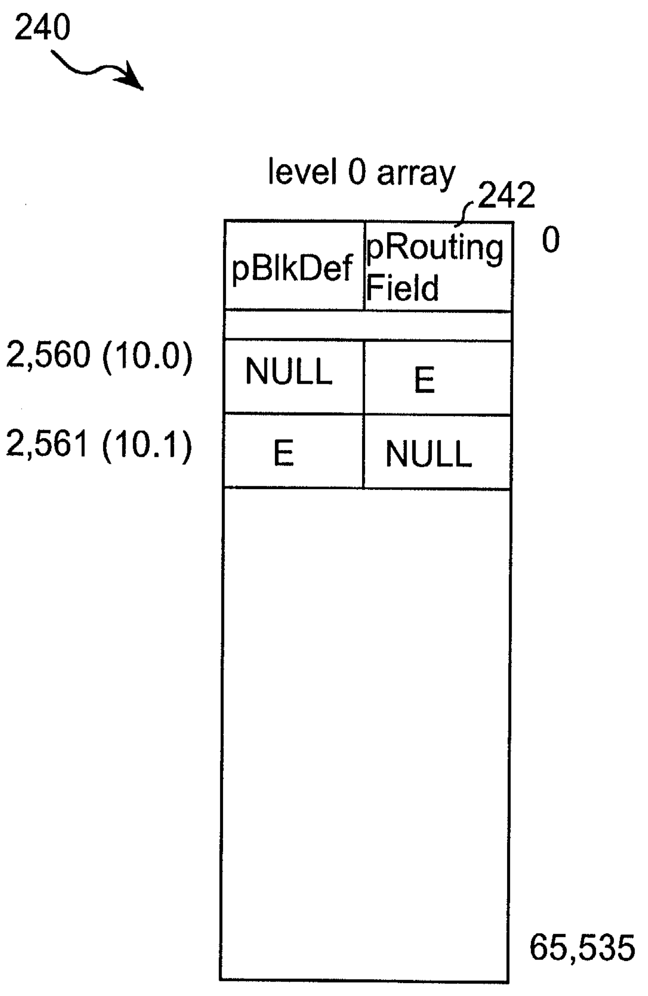


FIG. 19A

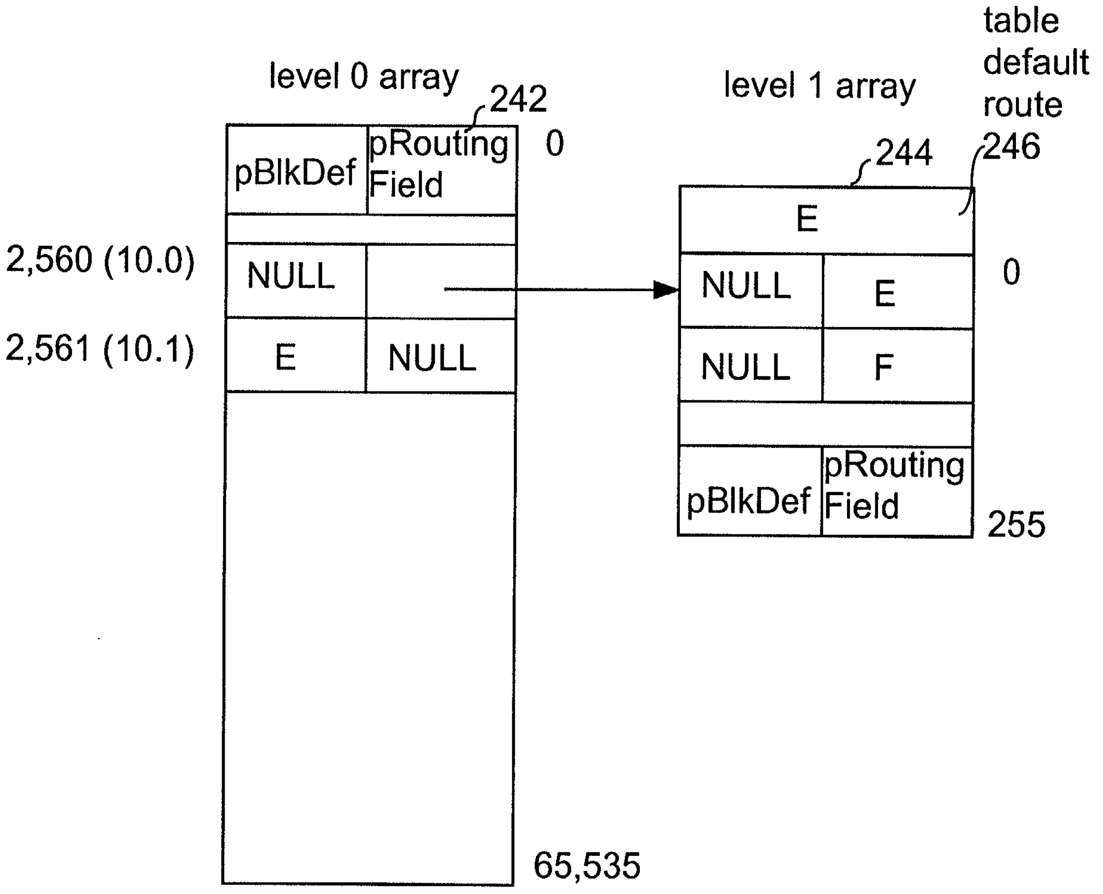
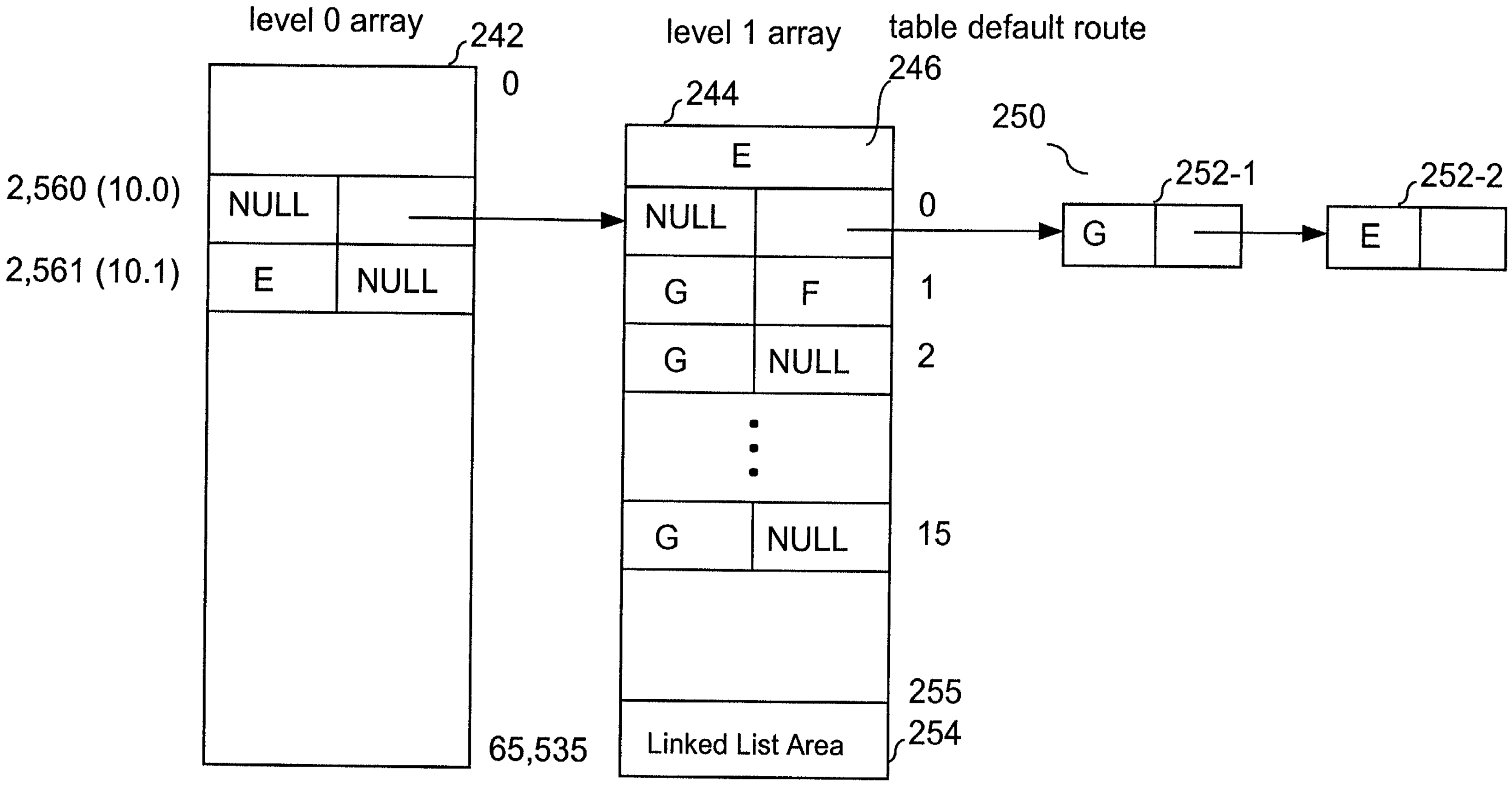


FIG. 19B



Overlapping Routes

FIG. 20

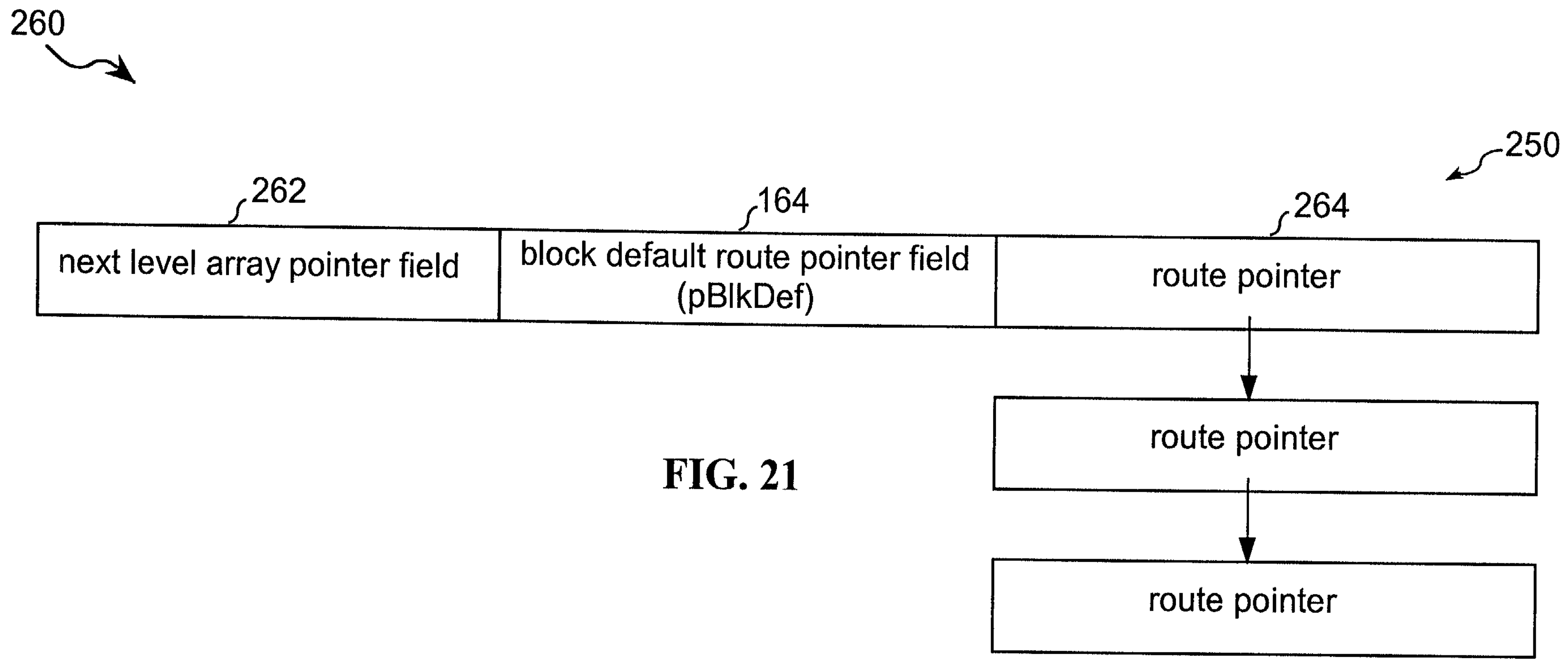

260
250
262
164
264
next level array pointer field
block default route pointer field
(pBlkDef)
route pointer
route pointer
route pointer

FIG. 21

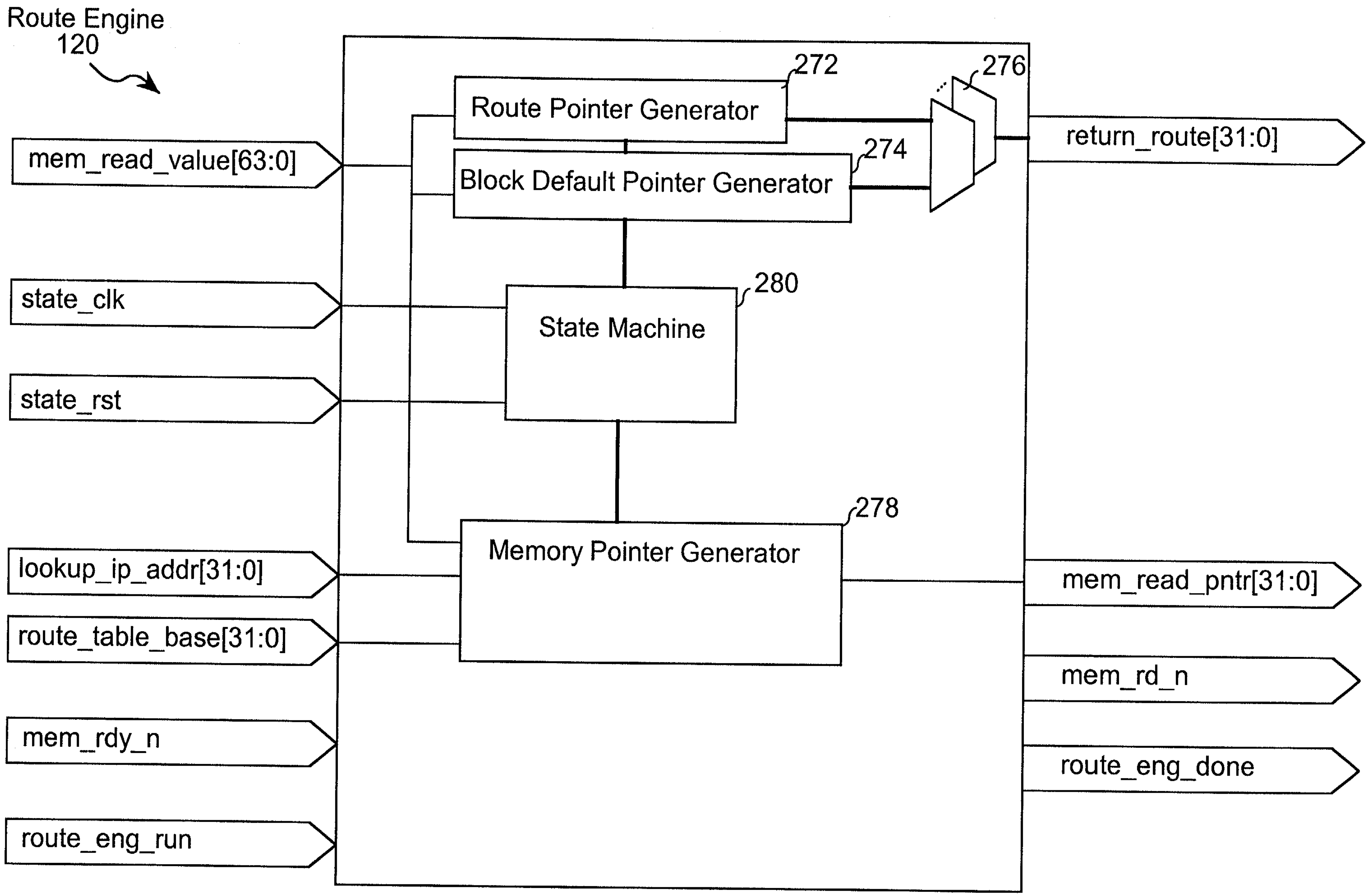

Route Engine
120
mem_read_value[63:0]
state_clk
state_rst
lookup_ip_addr[31:0]
route_table_base[31:0]
mem_rdy_n
route_eng_run
Route Pointer Generator
272
Block Default Pointer Generator
274
276
State Machine
280
Memory Pointer Generator
278
return_route[31:0]
mem_read_pntr[31:0]
mem_rd_n
route_eng_done

FIG. 22

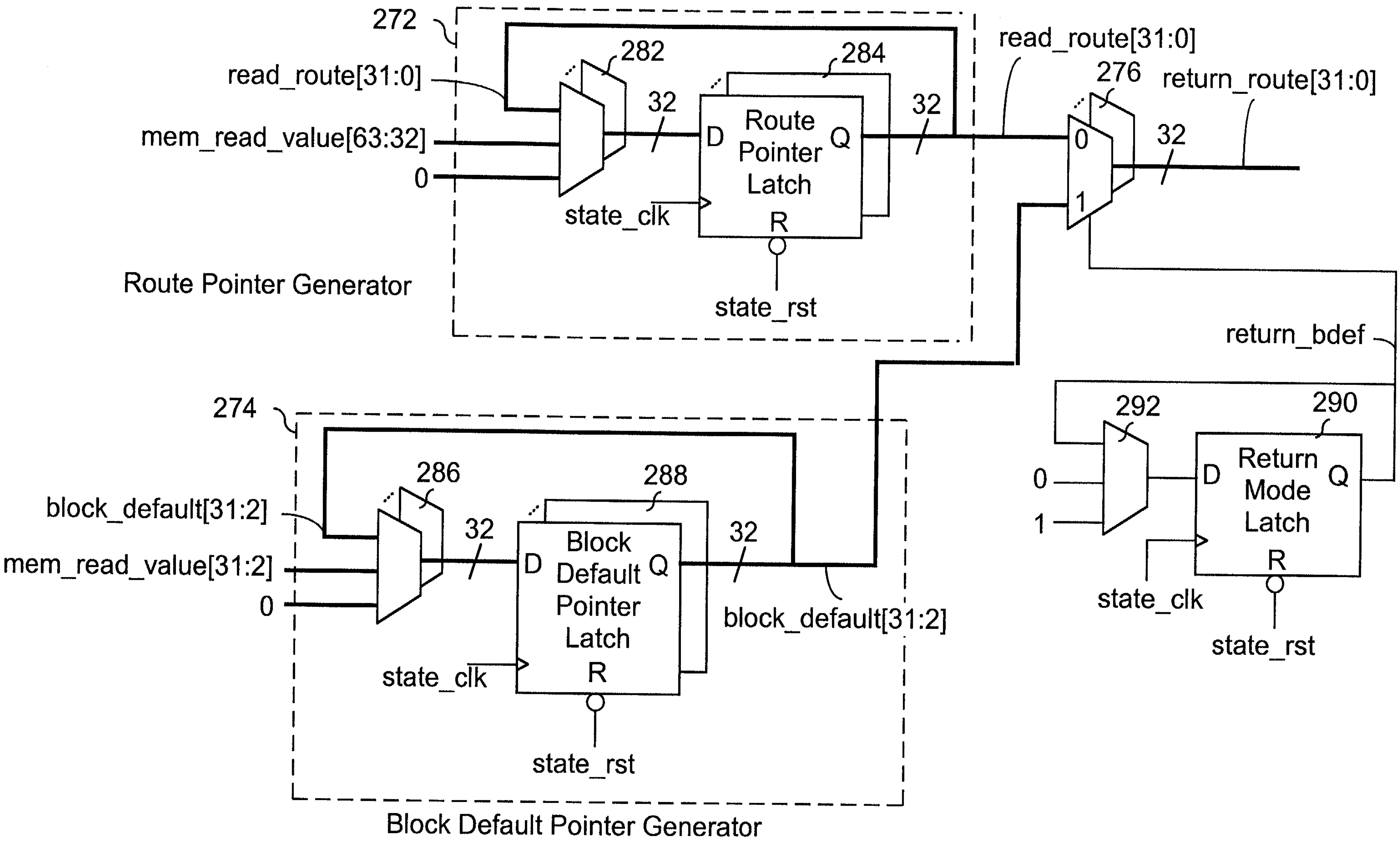

272
read_route[31:0]
mem_read_value[63:32]
0
282
32
284
Route Pointer Latch
D
Q
R
state_clk
state_rst
32
Route Pointer Generator
read_route[31:0]
276
0
1
32
return_route[31:0]
return_bdef
274
block_default[31:2]
mem_read_value[31:2]
0
286
32
288
Block Default Pointer Latch
D
Q
R
state_clk
state_rst
32
block_default[31:2]
Block Default Pointer Generator
292
0
1
290
Return Mode Latch
D
Q
R
state_clk
state_rst

FIG. 23

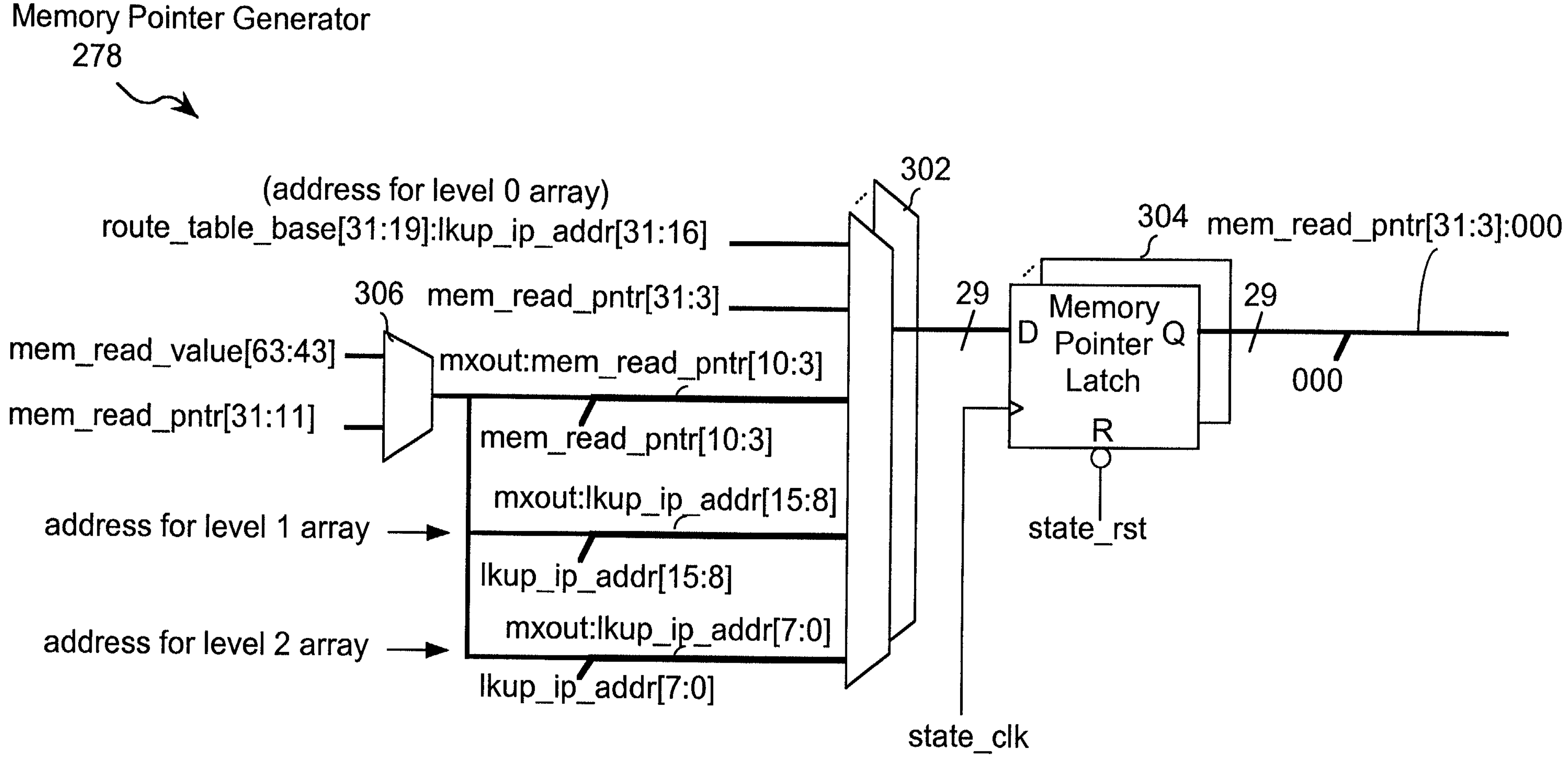

Memory Pointer Generator
278
(address for level 0 array)
route_table_base[31:19]:lkup_ip_addr[31:16]
302
304
mem_read_pntr[31:3]:000
306
mem_read_pntr[31:3]
29
Memory
Pointer
Latch
D
Q
R
29
mem_read_value[63:43]
mxout:mem_read_pntr[10:3]
000
mem_read_pntr[31:11]
mem_read_pntr[10:3]
mxout:lkup_ip_addr[15:8]
address for level 1 array
state_rst
lkup_ip_addr[15:8]
mxout:lkup_ip_addr[7:0]
address for level 2 array
lkup_ip_addr[7:0]
state_clk

FIG. 24

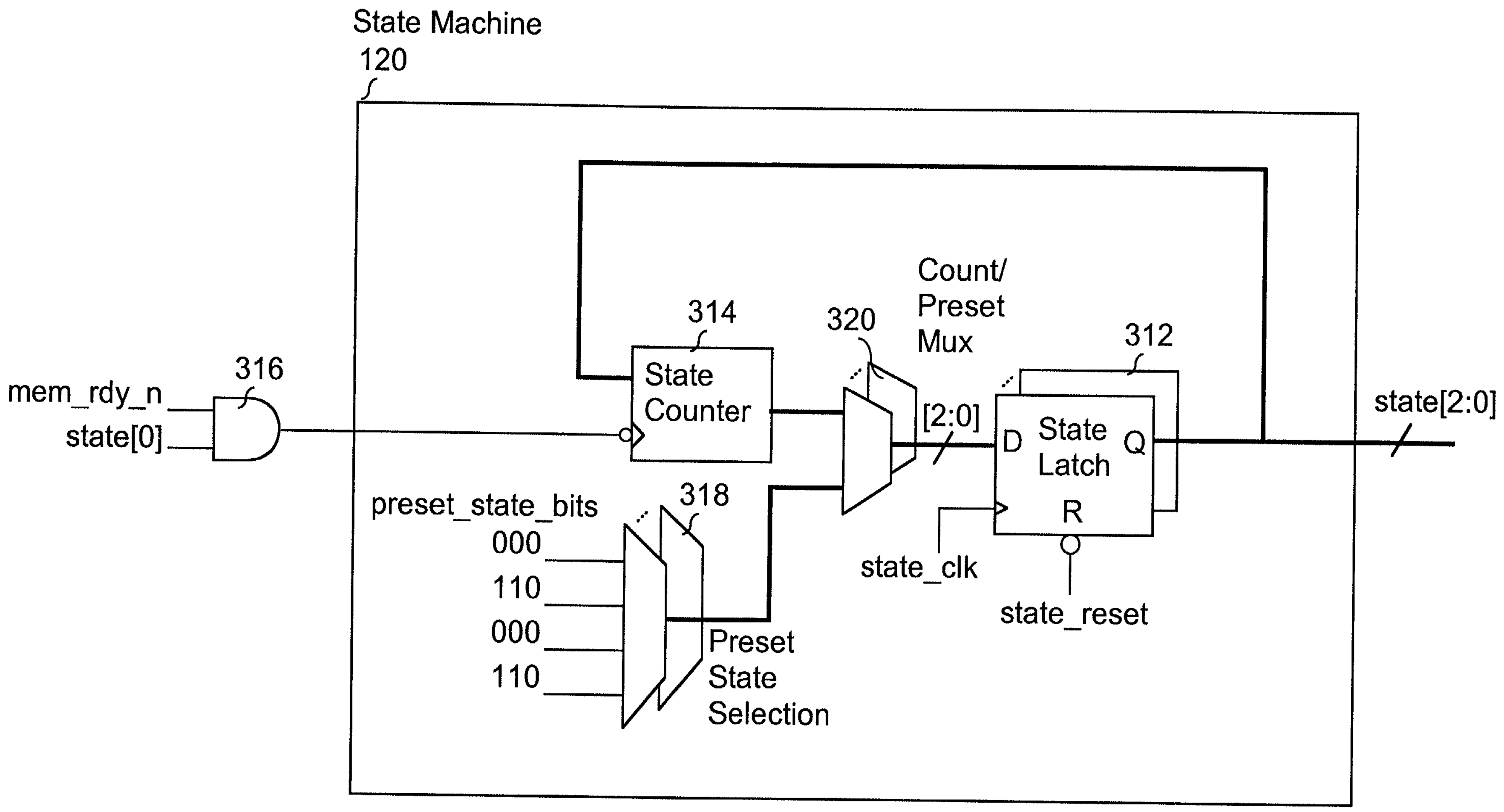

State Machine
120
mem_rdy_n
state[0]
316
314
State
Counter
320
Count/
Preset
Mux
312
[2:0]
D
State
Latch
Q
R
state_clk
state_reset
state[2:0]
preset_state_bits
000
110
000
110
318
Preset
State
Selection

FIG. 25

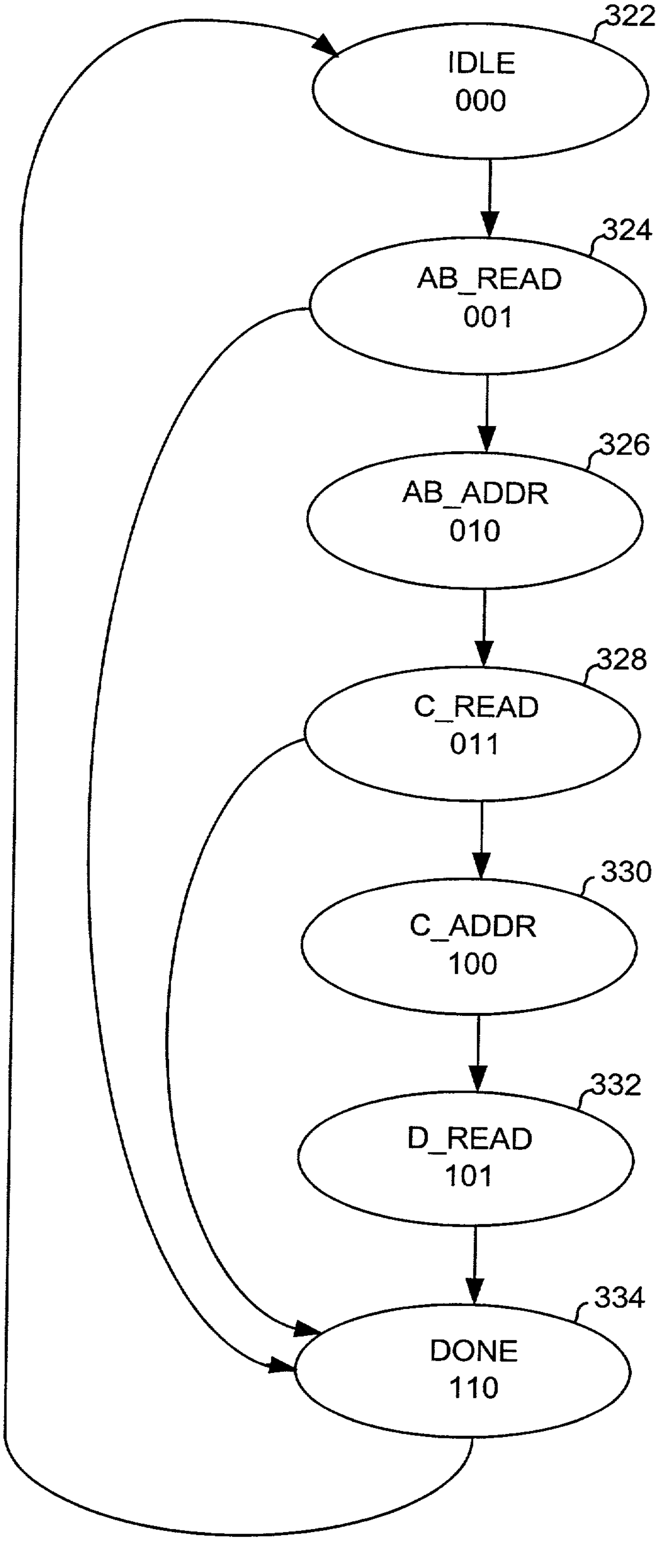

322
IDLE
000
324
AB_READ
001
326
AB_ADDR
010
328
C_READ
011
330
C_ADDR
100
332
D_READ
101
334
DONE
110

FIG. 26